IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appellant(s)	: Julie A Kadashevich	Examiner: Addy, Thjuan Knowlin
Serial No.	: 10/760,960	Group Art Unit: 2614
Filed	: 1/20/2004	
Atty Docket Client Ref.	: :	260-079 LOT9-2003-0119

Title : METHOD AND SYSTEM FOR MONITORING OFF-SCHEDULE SOFTWARE AGENTS

Mail Stop Appeal Brief-Patents
Commissioner for Patents
Alexandria, VA 22313-1450

APPEAL BRIEF

The Notice of Appeal was filed on February 5, 2010, and this Appeal Brief is responsive to the Final Office Action dated August 6, 2009.

I. Real Party in Interest

The real party in interest is International Business Machines Corporation.

II. Related Appeals and Interferences

Appellant is not aware of any appeals or interferences that are related to the present case.

III. Status of the Claims

This is an Appeal Brief from a decision dated August 6, 2009, finally rejecting all the claims currently pending in the present application. No claims have been allowed. The currently pending claims are 1-22, 24 and 25.

The status of claims 1-22, 24 and 25 is rejected.

The status of claim 23 is cancelled.

The rejection of claims 1-22, 24 and 25 under 35 U.S.C. 103 are the subject of this appeal.

IV. Status of Amendments

The claims were last amended in the response filed May 6, 2009, and those claim amendments have been entered. No amendments have been made in response to the Final Office Action of August 6, 2009.

A Notice of Appeal was filed on February 5, 2010.

V. Summary of Claimed Subject Matter

Claim 1 sets forth a method for identifying an off-schedule software agent operating in a computer system, said method comprising:

associating an entry time with said software agent entering a queue, wherein said queue is a run queue (see for example Run Queue 106 in Fig. 1) in which said software agent is stored in said computer system until an executive process (see for example Executive Process 108 in Fig. 1) in said computer system is free to process said software agent by running said software agent until said software agent is finished executing, wherein said entry time is a time at which a manager process moves said software agent from a holding queue to said run queue (see for example step 210 in Fig. 2, page 7, line 21 through line 2 on page 8, lines 22-25 on page 9, and lines 11-15 on page 10 in the Specification);

obtaining a clock signal associated with a clock time at which said software agent is still stored in said run queue (see for example step 310 in Fig. 3, and page 10 lines 24-25);

comparing said entry time to said clock time to obtain a queue time for said software agent (see for example page 10 lines 25-26 and/or step 308 in Fig. 3A);

comparing said queue time to a threshold limit (see for example page lines 26-27 and/or step 314 in Fig. 3A); and

identifying said software agent as said off-schedule software agent if said queue time exceeds said threshold time limit (see for example page 10 lines 27-28 and/or step 316 in Fig. 3A).

Claim 2 sets forth the method of claim 1, wherein said clock signal is obtained from a system clock (see for example page 10 line 25 and/or element 310 in Fig. 3A).

Claim 3 sets forth the method of claim 1, wherein said clock time indicates the current time (see for example page 10 line 24 and/or element 310 in Fig. 3A).

Claim 4 sets forth the method of claim 1, wherein said threshold time limit is associated with a graded scale for denoting the status of said software agent (see for example page 11 line 22 through line 2 on page 12 and lines 7-10 on page 19).

Claim 5 sets forth the method of claim 1, wherein said threshold time limit is specified by said computer system (see for example page 10 lines 19-21).

Claim 6 sets forth the method of claim 1, wherein said software agent is released from said queue if said queue time exceeds said threshold time limit (see for example page 11 lines 7-8 and/or element 320 in Fig. 3A).

Claim 7 sets forth the method of claim 1, wherein said software agent has a priority associated therewith (see for example page 11 lines 3-4).

Claim 8 sets forth the method of claim 7, wherein said priority is changed if said software agent is identified (see for example page 11 lines 3-5 and/or element 318 in Fig. 3A).

Claim 9 sets forth the method of claim 1, wherein said software agent has information associated therewith, said information allowing statistics of said software agent to be generated (see for example page 11 lines 8-9 and/or element 322 in Fig. 3A).

Claim 10 sets forth the method of claim 9, wherein said statistics of said software agent are compared to statistics associated with other software agents operating in said queue (see for example page 11 lines 9-10).

Claim 11 sets forth the method of claim 9, wherein at least a portion of said information is displayed to a user (see for example page 11 lines 11-12).

Claim 12 sets forth a method for managing a plurality of off-schedule software agents concurrently operating in a queue on a computer system, each of said plurality of software agents having data associated therewith, said method comprising:

receiving said data (see for example page 12 line 4 and/or step 326 in Fig. 3B);

processing said data to determine if any of said plurality of off-schedule software agents have excessive queue times, those of said plurality having excessive queue times identified as late software agents, wherein said excessive queue times are determined responsive to a run queue (see for example Run Queue 106 in Fig. 1) in which said plurality of software agents are stored in said computer system until executive processes (see for example Executive Processes 108 in Fig. 1) in said computer system are free to process respective ones of said plurality of off-schedule software agents by running said off-schedule software agents until said off-schedule software agents are each finished executing, wherein said off-schedule software agents are each determined to be off-schedule responsive to comparing differences between entry times at which a manager process moved each of said off-schedule software agents from a holding queue to said run queue and a later time at which said off-schedule software agents are still stored in said run queue with a threshold time limit associated with said run queue and determining that said differences each exceed said threshold time limit (see for example step 328 in Fig. 3B, lines 3-5 on page 12, line 21 through line 2 on page 8, lines 22-25 on page 9, and lines 11-15 on page 10); and

operating on at least said late software agents (see for example page 12 lines 5-14 and/or steps 330-340 in Fig. 3B).

Claim 13 sets forth the method of claim 12, wherein said operating further comprises:

determining if said late software agents reside in the same database (see for example page 12 lines 5-6 and/or step 332 in Fig. 3B).

Claim 14 sets forth the method of claim 13, further comprising parsing said late software agents across a plurality of databases (see for example page 12 lines 6-8 and/or step 334 in Fig. 3B).

Claim 15 sets forth the method of claim 12, wherein said threshold time limit is for determining the number of concurrently running software agents allowed to operate in said queue (see for example page 12 lines 8-11 and/or step 336 in Fig. 3B).

Claim 16 sets forth the method of claim 15, wherein the number of said software agents making up said plurality is compared to said threshold time limit (see for example page 12 lines 8-15 and/or 336 in Fig. 3B).

Claim 17 sets forth the method of claim 16, further comprising:

providing a plurality of executive processes if said plurality exceeds said threshold time limit when said comparison is made (see for example page 12 lines 12-13 and/or step 340 in Fig. 3B).

Claim 18 sets forth the method of claim 12, further comprising:

receiving said data associated with said off-schedule software agents from said run queue to produce received data (see for example page 11 lines 8-9 and/or step 322 in Fig. 3A);

defining criteria to be used with said received data (see for example page 11 line 16);

sorting said received data according to said criteria (see for example page 11 lines 17-20);

generating a list containing said received data (see for example page 11 lines 17-20);

filtering said received data (see for example page 11 lines 17-20); and

providing said received data to a document (see for example page 16 lines 13-24 and/or table 704 in Fig. 7A).

Claim 19 sets forth the method of claim 18, wherein said list is a sorted linked list (see for example page 15 lines 4-8, and/or lines 16-17 on page 16).

Claim 20 sets forth the method of claim 19, wherein said filtering removes unwanted agent data (see for example page 11 lines 13-16).

Claim 21 sets forth the method of claim 20, wherein said document is made available to a user (see for example page 11 line 16 and/or Fig. 7A).

Claim 22 sets forth the method of claim 21, wherein said document comprises:

instructions for said user to improve operation of at least one of said plurality of software agents (see for example lines 19-21 on page 16 and/or Recommendation 712 in Fig. 7A).

Claim 24 sets forth a computer system (see for example element 500 in Fig. 5) including at least one processor (see for example 502 in Fig. 5) and a computer readable memory (see for example 504, 506 and/or 508 in Fig. 5),said computer readable memory

having program code stored thereon for identifying an off-schedule software agent operating in a computer system, said program code comprising:

program code for associating an entry time with said software agent entering a queue, wherein said queue is a run queue in which said software agent is stored in said computer system until an executive process in said computer system is free to process said software agent by running said software agent until said software agent is finished executing, wherein said entry time is a time at which a manager process moves said software agent from a holding queue to said run queue (see for example step 210 in Fig. 2, page 7, line 21 through line 2 on page 8, lines 22-25 on page 9, and lines 11-15 on page 10 in the Specification);

program code for obtaining a clock signal associated with a clock time at which said software agent is still stored in said run queue (see for example step 310 in Fig. 3, and page 10 lines 24-25);

program code for comparing said entry time to said clock time to obtain a queue time for said software agent (see for example page 10 lines 25-26 and/or step 308 in Fig. 3A);

program code for comparing said queue time to a threshold limit (see for example page lines 26-27 and/or step 314 in Fig. 3A); and

program code for identifying said software agent as said off-schedule software agent if said queue time exceeds said threshold time limit (see for example page 10 lines 27-28 and/or step 316 in Fig. 3A).

Claim 25 sets forth a computer program product, comprising:

a computer readable memory (see for example Data Storage Device 508 in Fig. 5), said computer readable memory having program code stored thereon for identifying an off-schedule software agent operating in a computer system, said program code comprising

program code for associating an entry time with said software agent entering a queue, wherein said queue is a run queue in which said software agent is stored in said computer system until an executive process in said computer system is free to process said software agent by running said software agent until said software agent is finished executing, wherein said entry time is a time at which a manager process moves said software agent from a holding queue to said run queue (see for example step 210 in Fig. 2, page 7, line 21 through line 2 on page 8, lines 22-25 on page 9, and lines 11-15 on page 10 in the Specification),

program code for obtaining a clock signal associated with a clock time at which said software agent is still stored in said run queue (see for example step 310 in Fig. 3, and page 10 lines 24-25),

program code for comparing said entry time to said clock time to obtain a queue time for said software agent (see for example page 10 lines 25-26 and/or step 308 in Fig. 3A),

program code for comparing said queue time to a threshold limit (see for example page lines 26-27 and/or step 314 in Fig. 3A), and

program code for identifying said software agent as said off-schedule software agent if said queue time exceeds said threshold time limit (see for example page 10 lines 27-28 and/or step 316 in Fig. 3A).

VI. Grounds of Rejection to be Reviewed on Appeal

A. Claims 1-22, 24 and 25 stand rejected as obvious under 35 U.S.C. 103 based on the combination of U.S. Patent Application Publication 2001/0029526 ("Yokoyama") and U.S. Patent 5,978,594 ("Bonnell").

VII. Argument

A. The Examiner has failed to establish a *prima facie* case of obviousness under 35 U.S.C. 103 in the rejection of independent claims 1, 12, 24 and 25 using the combination of Yokoyama and Bonnell.

To establish *prima facie* obviousness of a claimed invention, all the claim limitations must be taught or suggested by the prior art. *In re Royka*, 490 F.2d 981, 180 USPQ 580 (CCPA 1974). "All words in a claim must be considered in judging the patentability of that claim against the prior art." *In re Wilson*, 424 F.2d 1382, 1385, 165 USPQ 494, 496 (CCPA 1970). Appellant respectfully asserts that the combination of Yokoyama and Bonnell does not disclose or suggest *associating an entry time with said software agent entering a queue, wherein said queue is a run queue in which said software agent is stored in said computer system until an executive process in said computer system is free to process said software agent by running said software agent until said software agent is finished executing, wherein said entry time is a time at which a manager process moves said software agent from a holding queue to said run queue*, obtaining a clock signal associated with *a clock time at which said software agent*

is still stored in said run queue, comparing said entry time to said clock time to obtain a queue time for said software agent [and] *comparing said queue time to a threshold limit, identifying said software agent as said off-schedule software agent if said queue time exceeds said threshold time limit*, as for example in the present independent claim 1.

U.S. Patent Application Publication 2001/0029526 ("Yokoyama"):

Yokoyama discloses traveling lists that are managed separately from agent programs. Traveling time is predicted by Yokoyama based on home terminal information at traveling destinations and information about programs to be executed at traveling destinations. The traveling destinations in Yokoyama are divided into multiple groups as needed and agent distribution is performed through these groups, to allow the time involved in traveling to be controlled.

A mobile agent execution state management program in Yokoyama monitors the traveling status of a mobile agent by referring to execution management data, which contains the execution state of a mobile agent. In cases such as when the mobile agent has not returned to a server from a number of home terminals significantly after a traveling limit time has passed, the Yokoyama server administrator is notified. A service schedule manager in Yokoyama refers to a service schedule, which contains the distribution schedule for services, and issues request events so that, for example, agents containing a particular service program are distributed at service distribution times.

Service program data is formed by Yokoyama from a set of service program-specific information, where there is one service program-specific information for each type of provided service. The service program-specific information of Yokoyama is formed from a service name, attributes (e.g., service provider name), an average execution time, an average memory usage, a service price, a distribution plan, a traveling limit time, and a program body data. The distribution plan of Yokoyama is used to determine if distribution times are to be strictly followed or if a certain amount of leeway should be given while keeping costs (e.g., communication fees) down.

The mobile agent execution state management data of Yokoyama is used to manage the execution state of a mobile agent that has been sent out from the server system. The Yokoyama agent execution state management data is formed from a mobile agent ID, a mobile agent state (e.g., traveling, traveling completed, fault processing), a list of service program names contained in the mobile agent, and traveling data. The traveling data in Yokoyama is formed from a traveling list, a send time, a scheduled end time, and an end time for the mobile agent, as well as fault data. The Yokoyama mobile agent distributor performs operations to manage the traveling state and fault state of a mobile agent based on mobile agent execution state management data.

When a fault message is sent from a home terminal in the Yokoyama system, a message processor uses the mobile agent execution state management data and the mobile agent identifier in the fault message to determine the mobile agent and the traveling list for which the fault is occurring. Using the traveling

list, a fault avoidance traveling list, in which the faulty home terminal is removed from the list, is generated by Yokoyama and sent to the home terminal that sent the fault message.

U.S. Patent 5,978,594 ("Bonnell"):

Bonnell discloses a system for managing a computer network, in which a manager software system is installed on a network management computer system within the network, and an agent software system is installed on each of the server computer systems in the network. A knowledge module text file is stored on the network manager computer system so that the manager software system can transmit knowledge to the various agent software systems throughout the network, for use by the agents in monitoring and managing the server on which they are installed. Bonnell teaches using high level interpretable script language programs in connection with the agent software systems for discovering resources on the network, monitoring aspects of resources, and taking recovery actions automatically in the event of an alarm condition.

Fig. 3 of Bonnell includes a run queue scheduler that maintains a list of runnable jobs or commands, together with the times at which they should be run and their desired frequency. Bonnell discloses that by checking a timer within agent software system, the run queue scheduler is capable of "waking up" at appropriate times to route runnable jobs or commands to a command execution manager.

Bonnell shows an initialization-type setup procedure in Fig. 7 in which the

manager software parses through a knowledge module, extracting information therefrom to create a knowledge database. The manager software accepts input from the user indicating types of computers that are present throughout the network, and are accordingly to be managed. The manager software searches the knowledge database to determine the knowledge therein pertinent to such computers, and sends messages to the various agent software systems throughout the network, containing knowledge and/or script that will be pertinent and useful for the agent software systems in carrying out their respective tasks in network management. Each agent software system receives the messages and parses the information to create a knowledge database of its own on the server on which it is installed. The agent software stores any script programs sent by the manager software system, and a knowledge database manager in the agent software system creates an agent knowledge database. The knowledge database manager in the agent software also determines whether the knowledge stored in the created knowledge database indicates that periodic monitoring procedures are to be executed relating to particular resources. If so, the knowledge database manager creates appropriate job descriptions and places them in a run queue so that a run queue scheduler may initiate those processes at the appropriate times.

Bonnell also includes a method, illustrated in Fig. 8, for discovering resources on a server computer system using a high-level interpretable language. Bonnell teaches that the discovery procedure may be initiated when the timer within agent software system indicates that a discovery procedure stored in run queue is ready to be executed, or when the manager software system sends a

message to the agent software system indicating that a discovery procedure should be executed.

Bonnell further describes a procedure for monitoring an aspect of a resource, as illustrated in Fig. 9. The monitoring procedure of Bonnell, like the discovery procedure of Fig. 8, may begin when a timer within the agent software indicates that the monitoring procedure should be executed, or when the manager software system sends a message to agent software system indicating that a monitoring procedure should be initiated.

Independent Claims 1, 12, 24 and 25:

Claim 1 sets forth a method for identifying an off-schedule software agent operating in a computer system, said method comprising:

> associating an entry time with said software agent entering a queue, ***wherein said queue is a run queue in which said software agent is stored in said computer system until an executive process in said computer system is free to process said software agent by running said software agent until said software agent is finished executing, wherein said entry time is a time at which a manager process moves said software agent from a holding queue to said run queue***;
> obtaining a clock signal associated with ***a clock time at which said software agent is still stored in said run queue***;
> comparing said entry time to said clock time to obtain a queue time for said software agent;
> comparing said queue time to a threshold limit; and
> identifying said software agent as said off-schedule software agent if said queue time exceeds said threshold time limit. (emphasis added)

The combination of Yokoyama and Bonnell results in a system that operates to calculate a predicted time for a traveling operation performed by a mobile agent that returns to a server system after having traversed one or more

remote home terminals (as in Yokoyama, see Fig. 16 and paragraphs 132 and 133) and that also includes a timer within agent software system that can be used to determine appropriate times at which runnable jobs or commands are to be routed to a command execution manager (as in Bonnell, see the periodic monitoring procedure described in Fig. 7 and lines 39-45 in column 7, the discovery procedure described in Fig. 8 and lines 48-53 in column 7, and/or the monitoring procedure described in Fig. 9 and lines 16-19 in column 7). Applicant notes that while a timer may be used by a software agent in Bonnell to start a discovery procedure already stored in a run queue (line 51, column 7), the time at which the discovery procedure was added to the run queue is not recorded. Similarly, Fig. 9 of Bonnell shows that a timer may be used to start execution of the monitoring process, but includes no mention of recording a time at which the monitoring process enters a run queue.

Accordingly, the combination of the timer used to start execution of jobs or commands in Bonnell and/or the travelling time predictions disclosed by Yokoyama fails to disclose or suggest any method that includes measuring a time that an agent has been stored in a run queue measured since an entry time at which an agent manager moves the agent from a holding queue to the run queue. Accordingly, the combination of Yokoyama and Bonnell does not disclose or suggest associating an entry time with said software agent entering a queue, wherein said queue is a run queue in which said software agent is stored in said computer system until an executive process in said computer system is free to process said software agent by running said software agent until said software

agent is finished executing, *wherein said entry time is a time at which a manager process moves said software agent from a holding queue to said run queue*, obtaining a clock signal associated with *a clock time at which said software agent is still stored in said run queue*, comparing said entry time to said clock time to obtain a queue time for said software agent *comparing said queue time to a threshold limit, identifying said software agent as said off-schedule software agent if said queue time exceeds said threshold time limit*, as in the present independent claim 1.

For the above reasons, it should also be evident that the combination of Yokoyama and Bonnell also does not disclose or suggest a method for managing a plurality of off-schedule software agents concurrently operating in a queue on a computer system, each of said plurality of software agents having data associated therewith, said method comprising . . . processing said data to determine if any of said plurality of off-schedule software agents have excessive queue times, those of said plurality having excessive queue times identified as late software agents, wherein said excessive queue times are determined responsive to a run queue in which said plurality of software agents are stored in said computer system until executive processes in said computer system are free to process respective ones of said plurality of off-schedule software agents by running said off-schedule software agents until said off-schedule software agents are each finished executing, *wherein said off-schedule software agents are each determined to be off-schedule responsive to comparing differences between entry times at which a manager process moved each of said off-schedule software agents from a holding*

queue to said run queue and a later time at which said off-schedule software agents are still stored in said run queue with a threshold time limit associated with said run queue and determining that said differences each exceed said threshold time limit . . ., as in the present independent claim 12.

The above should also make evident that the combination of Yokoyama and Bonnell also does not disclose or suggest a computer system including at least one processor and a computer readable memory, said computer readable memory having program code stored thereon for identifying an off-schedule software agent operating in a computer system, said program code comprising: *program code for associating an entry time with said software agent entering a queue, wherein said queue is a run queue in which said software agent is stored in said computer system until an executive process in said computer system is free to process said software agent by running said software agent until said software agent is finished executing, wherein said entry time is a time at which a manager process moves said software agent from a holding queue to said run queue; program code for obtaining a clock signal associated with a clock time at which said software agent is still stored in said run queue; program code for comparing said entry time to said clock time to obtain a queue time for said software agent; program code for comparing said queue time to a threshold limit; and program code for identifying said software agent as said off-schedule software agent if said queue time exceeds said threshold time limit*, as in the present independent claim 24.

And the above should further make evident that the combination of Yokoyama and Bonnell also does not disclose or suggest A computer program product, comprising: a computer readable memory, said computer readable memory having program code stored thereon for identifying an off-schedule software agent operating in a computer system, said program code comprising *program code for associating an entry time with said software agent entering a queue, wherein said queue is a run queue in which said software agent is stored in said computer system until an executive process in said computer system is free to process said software agent by running said software agent until said software agent is finished executing, wherein said entry time is a time at which a manager process moves said software agent from a holding queue to said run queue, program code for obtaining a clock signal associated with a clock time at which said software agent is still stored in said run queue, program code for comparing said entry time to said clock time to obtain a queue time for said software agent, program code for comparing said queue time to a threshold limit, and program code for identifying said software agent as said off-schedule software agent if said queue time exceeds said threshold time limit*, as in the present claim 25.

For the above reasons, Appellant respectfully urges that the combination of Yokoyama and Bonnell does not disclose or suggest all of the features of the present independent claims 1, 12, 24 and 25. Accordingly, the combination of Yokoyama and Bonnell does not support a *prima facie* case of obviousness under 35 U.S.C. 103 with regard to the present independent claims. As to the remaining claims 2-11 and 13-22, they each depend from the independent claims 1 and 12,

and are respectfully believed to be patentable over the combination of Yokoyama and Bonnell for at least the same reasons.

Conclusion

Appellant respectfully submits that the rejections of the present claims under 35 U.S.C. 103 are improper for at least the reasons set forth above. Appellants accordingly request that the rejections be withdrawn and the pending claims be allowed.

Respectfully submitted,

INTERNATIONAL BUSINESS MACHINES CORPORATION

By: /David Dagg/
David A. Dagg
Reg. No. 37,809
Attorney for Assignee

Date: July 2, 2010

David A. Dagg – Patent Attorney, P.C.
44 Chapin Road
Newton MA 02459
(617) 630-1131

VIII. Claims Appendix

The currently pending claims are as follows:

1. (previously presented) A method for identifying an off-schedule software agent operating in a computer system, said method comprising:

associating an entry time with said software agent entering a queue, wherein said queue is a run queue in which said software agent is stored in said computer system until an executive process in said computer system is free to process said software agent by running said software agent until said software agent is finished executing, wherein said entry time is a time at which a manager process moves said software agent from a holding queue to said run queue;

obtaining a clock signal associated with a clock time at which said software agent is still stored in said run queue;

comparing said entry time to said clock time to obtain a queue time for said software agent;

comparing said queue time to a threshold limit; and

identifying said software agent as said off-schedule software agent if said queue time exceeds said threshold time limit.

2. (original) The method of claim 1, wherein said clock signal is obtained from a system clock.

3. (original) The method of claim 1, wherein said clock time indicates the current time.

4. (previously presented) The method of claim 1, wherein said threshold time limit is associated with a graded scale for denoting the status of said software agent.

5. (original) The method of claim 1, wherein said threshold time limit is specified by said computer system.

6. (previously presented) The method of claim 1, wherein said software agent is released from said queue if said queue time exceeds said threshold time limit.

7. (previously presented) The method of claim 1, wherein said software agent has a priority associated therewith.

8. (previously presented) The method of claim 7, wherein said priority is changed if said software agent is identified.

9. (previously presented) The method of claim 1, wherein said software agent has information associated therewith, said information allowing statistics of said software agent to be generated.

10. (previously presented) The method of claim 9, wherein said statistics of said software agent are compared to statistics associated with other software agents operating in said queue.

11. (original) The method of claim 9, wherein at least a portion of said information is displayed to a user.

12. (previously presented) A method for managing a plurality of off-schedule software agents concurrently operating in a queue on a computer system, each of said plurality of software agents having data associated therewith, said method comprising:

receiving said data;

processing said data to determine if any of said plurality of off-schedule software agents have excessive queue times, those of said plurality having excessive queue times identified as late software agents, wherein said excessive queue times are determined responsive to a run queue in which said plurality of software agents are stored in said computer system until executive processes in said computer system are free to process respective ones of said plurality of off-schedule software agents by running said off-schedule software agents until said off-schedule software agents are each finished executing, wherein said off-schedule software agents are each determined to be off-schedule responsive to comparing differences between entry times at which a manager process moved each of said off-schedule software agents from a holding queue to said run queue and a later time at which said off-schedule software agents are still stored in

said run queue with a threshold time limit associated with said run queue and determining that said differences each exceed said threshold time limit; and

operating on at least said late software agents.

13. (previously presented) The method of claim 12, wherein said operating further comprises:

determining if said late software agents reside in the same database.

14. (previously presented) The method of claim 13, further comprising parsing said late software agents across a plurality of databases.

15. (previously presented) The method of claim 12, wherein said threshold time limit is for determining the number of concurrently running software agents allowed to operate in said queue.

16. (previously presented) The method of claim 15, wherein the number of said software agents making up said plurality is compared to said threshold time limit.

17. (original) The method of claim 16, further comprising:

providing a plurality of executive processes if said plurality exceeds said threshold time limit when said comparison is made.

18. (previously presented) The method of claim 12, further comprising:

receiving said data associated with said off-schedule software agents from said run queue to produce received data, ;

defining criteria to be used with said received data;

sorting said received data according to said criteria;

generating a list containing said received data;

filtering said received data; and

providing said received data to a document.

19. (original) The method of claim 18, wherein said list is a sorted linked list.

20. (original) The method of claim 19, wherein said filtering removes unwanted agent data.

21. (original) The method of claim 20, wherein said document is made available to a user.

22. (previously presented) The method of claim 21, wherein said document comprises:

instructions for said user to improve operation of at least one of said plurality of software agents.

23. (cancelled)

24. (previously presented) A computer system including at least one processor and a computer readable memory,

said computer readable memory having program code stored thereon for identifying an off-schedule software agent operating in a computer system, said program code comprising:

program code for associating an entry time with said software agent entering a queue, wherein said queue is a run queue in which said software agent is stored in said computer system until an executive process in said computer system is free to process said software agent by running said software agent until said software agent is finished executing, wherein said entry time is a time at which a manager process moves said software agent from a holding queue to said run queue;

program code for obtaining a clock signal associated with a clock time at which said software agent is still stored in said run queue;

program code for comparing said entry time to said clock time to obtain a queue time for said software agent;

program code for comparing said queue time to a threshold limit; and

program code for identifying said software agent as said off-schedule software agent if said queue time exceeds said threshold time limit.

25. (previously presented) A computer program product, comprising:

a computer readable memory, said computer readable memory having program code stored thereon for identifying an off-schedule software agent operating in a computer system, said program code comprising

program code for associating an entry time with said software agent entering a queue, wherein said queue is a run queue in which said software agent

is stored in said computer system until an executive process in said computer system is free to process said software agent by running said software agent until said software agent is finished executing, wherein said entry time is a time at which a manager process moves said software agent from a holding queue to said run queue,

program code for obtaining a clock signal associated with a clock time at which said software agent is still stored in said run queue,

program code for comparing said entry time to said clock time to obtain a queue time for said software agent,

program code for comparing said queue time to a threshold limit, and

program code for identifying said software agent as said off-schedule software agent if said queue time exceeds said threshold time limit.

IX. Evidence Appendix

None.

X. Related Proceedings Appendix

None.